LINKgroup

Distance Learning System

Perzistentnost

Python data Access and Processing

vladimir.maric@link.co.rs

Uvod u perzistentnost

- Tokom izvršavanja programa brojni objekti bivaju kreirani. Svi oni postoje u radnoj memoriji onoliko dugo koliko su potrebni programu za nesmetano izvršavanje. Kada aplikacija završi svoje izvršavanje, oni se trajno gube

- Postoje brojni načini da jedna aplikacija sačuva određene podatke, a da pri tom to nije čuvanje u radnoj memoriji

- Kako sačuvati ove objekte i nakon završetka izvršavanja aplikacije? Odgovor leži u **perzistenciji**.

Šta je perzistencija?

- Pojam perzistencije se odnosi na mogućnost čuvanja objekta i nakon prestanka izvršavanja aplikacije.
- Gotovo sve aplikacija imaju potrebu da objekat koji je kreiran sačuvaju za kasnije pristupanje i eventualno modifikovanje. Aplikacija može baratati podacima o zaposlenima, konfiguracionim podacima aplikacije, podacima o statistici korišćenja. Potrebe su neograničene.
  - Tranzijentan objekat
  - Perzistentan objekat

Persistent VS Transient

```
person = Person()
```

- Instanciranjem, objekat postaje transijentan objekat u skupu objekata koje aplikacija koristi. To se može reći i za objekat person, čije je instanciranje upravo prikazano.
- Da bi objekat postao perzistentan potrebno ga je na neki način sačuvati i učiniti dostupnim i nakon prestanka izvršavanja aplikacije.
- Perzistentni objekti omogućavaju hibernaciju aplikacije, ali i razmenu informacija između aplikacija

Smeštanje objekata u fajl
(pdap-ex03 persistenceintro)

- Objekat je moguće učiniti perzistentnim jednostavnim pisanjem u fajl i čitanjem fajla

Upis

```
file = open("person","w")
file.write(f"{self.name},{self.age},{self.pid}")
file.close()
```

Čitanje

```
file = open("person","r")
line = file.readline().split(",")
file.close()
return Person(line[0],line[1],line[2])
```

JSON serijalizacija
(pdap-ex03 jsonpersist)

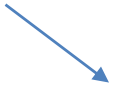
```
import json
```

- Najčešće se sadržaj prilikom upisa u fajl serijalizuje, a prilikom preuzimanja iz fajla deserijalizuje u neki od standardnih formata
- Standardni formati za serijalizaciju su: XML, JSON, YAML, CSV i drugi
- Modul **json** sadrži podršku za JSON serijalizaciju ugrađenih tipova

```
data = [
    {"name":"TV","price":350.55},
    {"name":"Phone","price":125.99}
]
json_string = json.dumps(data)
```

```
[
    {"name":"TV","price":350.55},
    {"name":"Phone","price":125.99}
]
```

```
data = json.loads(json_string)
```



[{"name": "TV", "price": 350.55}, {"name": "Phone", "price": 125.99}]

Binarna serijalizacija
(pdap-ex03 binpersist)

```
import pickle
```

- Python elemente je moguće sačuvati binarno kroz serijalizator **pickle**
- Pickle pretvara objekte u niz bajtova (**bytes**)

```
p = Person("Peter")
bin_data = pickle.dumps(p)
```

```
p = Person("Sally")
bin_data = pickle.dump(
    p,open("person.dat","wb")
)
```

```
person = pickle.load(
    open("person.dat","rb")
)
print("Hello",person.username)
```

→ `Hello Sally`

00101010101010010101010101010

XML serijalizacija
(pdap-ex03 xmlpersist)

```
import xml.etree.ElementTree as xml
```

- XML je pored JSON-a najpopularniji jezik za čuvanje podataka u tekstualnom formatu

```
p = Person("Peter")

root = xml.Element("person")
name = xml.SubElement(root,"name")
name.text = p.username
tree = xml.ElementTree(root)

tree.write("person.xml")
```

```
tree = xml.parse(
    open("person.xml","r")
)
p = Person(tree.getroot()[0].text)
print(p.username)
```

→ `Peter`

<person><name>Peter</name></person>

Baza podataka

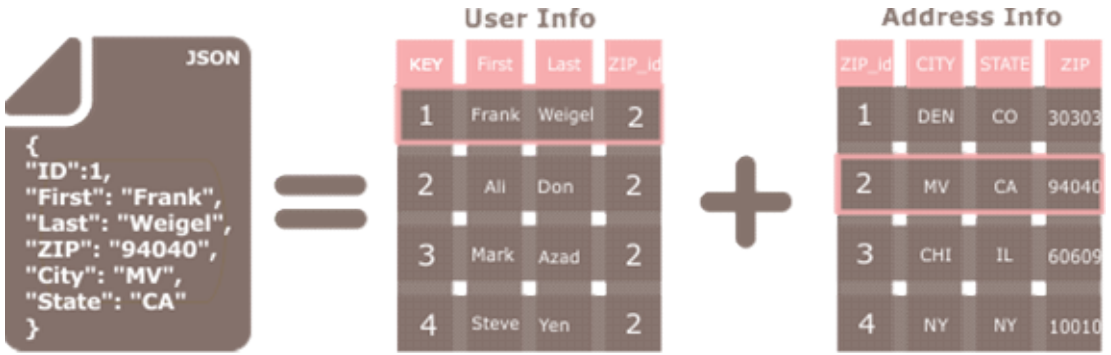
- Ultimativno rešenje za realizovanje perzistencije jeste korišćenje baza podataka i sistema za upravljanje bazama podataka
- Postoje razne vrste baza podataka, u zavisnosti od načina na koji su podaci organizovani i na koji im se pristupa. Do sada predstavljene su relacione baze podataka kod kojih se podaci smeštaju u tabele koje su povezane određenim relacijama. Trenutno, najrasprostranjeniji tip baza podataka jeste **relacioni**

NoSQL

- NoSQL je relativno nova kovanica, koja u poslednje vreme privlači sve veću pažnju. Najpravilnije je reći da ona predstavlja skraćenicu za pojam *Not Only SQL*.
- Reč je o mehanizmima za baratanje podacima koji su modelovani tako da ne koriste tabele i njihove relacije i zavisnosti. Jednostavno, određeni scenariji, tj. primene određenih aplikacija bolje se snalaze sa drugačijim mehanizmima skladištenja podataka
- Postoji nekoliko tipova sistema za upravljanje bazama podataka koji se mogu smatrati NoSQL bazama. Podela se zasniva na samom načinu smeštanja podataka.

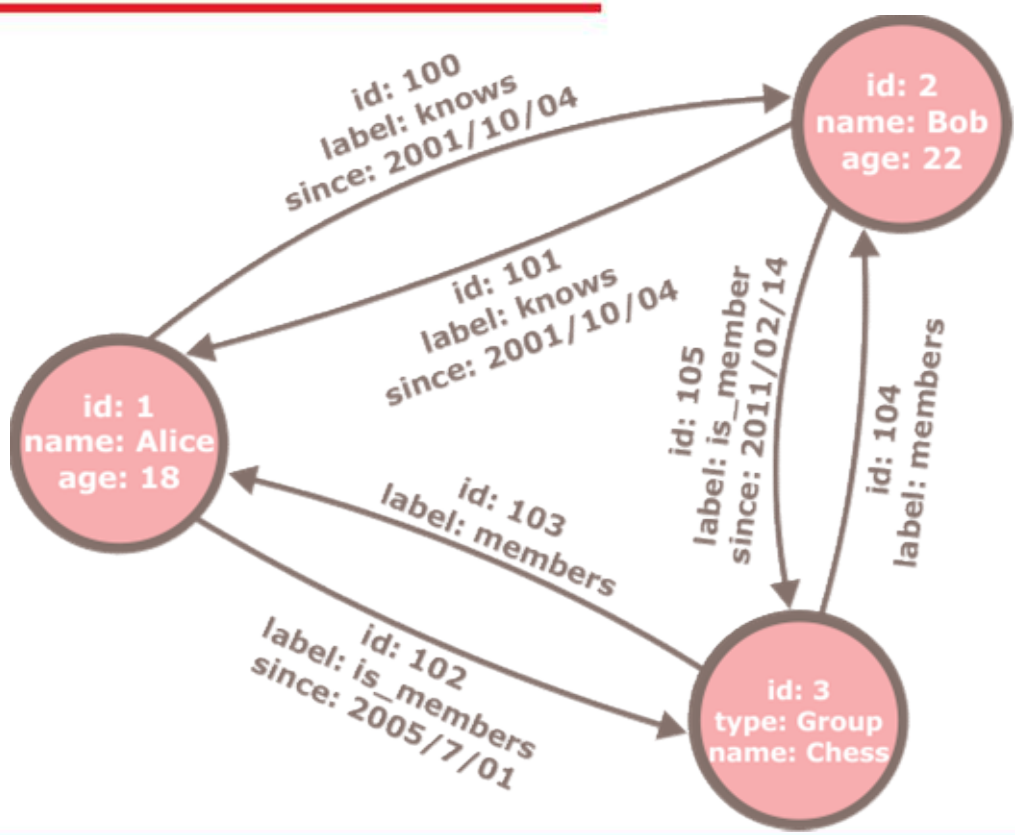

Document

- Osnova ovakvog tipa skladištenja podataka su, naravno, dokumenti. Dokument enkapsulira podatke u nekoj standardnoj formi. U tu svrhu najčešće se koriste XML, YAML i JSON.
- Dokumenti se u bazi adresiraju preko jedinstvenih ključeva koji predstavljaju dokument.
- Ovi dokumenti se mogu po analogiji uporediti sa redovima relacionih baza podataka. Ipak, tabele relacionih sistema uvek sadrže podatke istih tipova, dok dokumenti mogu imati proizvoljan skup podataka, koji može biti potpuno jedinstven na nivou jednog dokumenta.
- Na je prikazan izgled jednog dokumenta ovakvih baza podataka, kao i analogni prikaz u relacionom sistemu.



Graph

- Ovakav tip baze podataka zasniva se na strukturi grafa, tj. matematičkoj teoriji grafa. Fokus je na strukturi podataka, tj. na njihovoj međusobnoj povezanosti. Koristi čvorove, veze i propertije za reprezentovanje i čuvanje podataka.
- Čvorovi predstavljaju entitete kao što su osobe, poslovi, računi ili bilo koji drugi entiteti koji odgovaraju potrebama sistema. Čvorovi su u međusobnim relacijama, tj. vezama u zavisnosti od potreba sistema.
- Veze, odnosno relacije mogu imati propertije u formi ključa i vrednosti, što takođe mogu imati i sami čvorovi. Ovi propertiji sadrže podatke koji bliže određuju čvor ili vezu na koju se odnose.

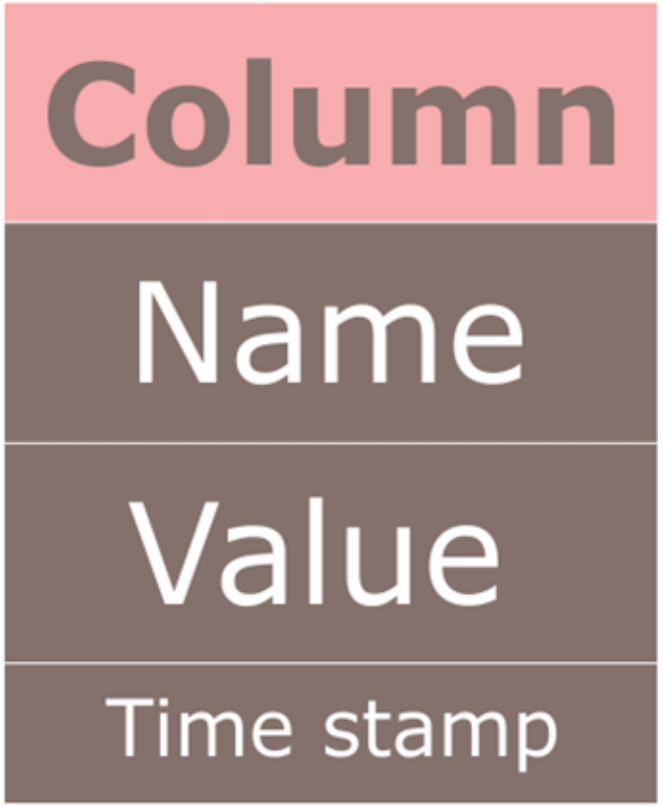


Key-value

- NoSQL baze podataka koje su najlakše za implementaciju su svakako baze koje smeštanje podataka zasnivaju na parovima ključeva i vrednosti. Jednostavno, podaci su smešteni kao kolekcije parova ključeva i vrednosti, tako da se u jednoj kolekciji može pojaviti samo jedan ključ.

Column

- Column su distribuirana skladišta podataka gde se podaci smeštaju u uređene parove ključeva i vrednosti kojim je pridodat i vremenski podatak. Ova tri podatka (naziv, vrednost i timestamp) zajedno čine kolonu (Column). Kolone se koriste za smeštanje vrednosti, a vremenski podatak se koristi kako bi se utvrdila relevantnost podataka. Ipak, ne sme se mešati kolona tabela relacionih baza podatka i kolona o kojoj je ovde reč.

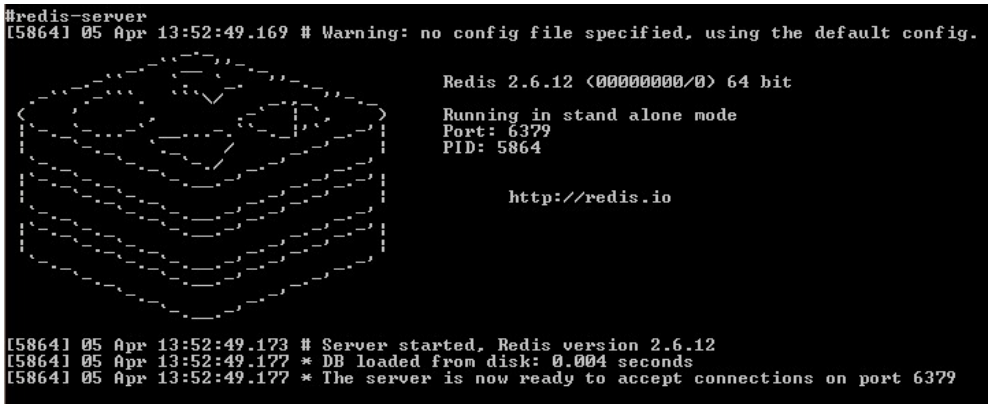


Redis (http://redis.io/)

- Redis je key-value baza podataka koja podatke čuva u memoriji
- Redis se najčešće koristi kao
  - Sistem za keširanje
  - Sistem za skladištenje sesija
  - Sistem za privremeno sinhronizaciju
  - Message queue-ing sistem

Upotreba redisa

- Redis zahteva runtime (server) da bi mogao biti korišćen



- Direktan konzolni pristup redis serveru vrši se pomoću alata **redis-cli**
- Ključevi u redisu se postavljaju / preuzimaju metodama **set** i **get**

```
#redis-cli
redis 127.0.0.1:6379> set marco polo
OK
redis 127.0.0.1:6379> get marco
"polo"
redis 127.0.0.1:6379>
```

Povezivanje redisa sa Python-om

- Biblioteka za rad sa redis bazom podataka dostupna je na pypi repozitorijumu

pip install redis

```
import redis

r = redis.Redis()
r.set("peter","Peter Jackson")
```

Perzistentnost objekata pomoću Redisa
(pdap-ex04 simpleredis.py)

- Redis ne može direktno prihvatiti objekte, ali se oni mogu serijalizovati prilikom smeštanja u Redis

```
user = {
    "firstname":"Peter",
    "lastname":"Jackson"
}
r.set("peter",json.dumps(user))
```

```
user = json.loads(r.get("peter"))
print(
    user["firstname"],
    user["lastname"]
)
```

Redis publisher subscriber
(pdap-ex04 redispubsub.py)

- Redis podržava publisher / subscriber model

```
r = redis.Redis()
r.publish("messages","Hello!")
```

```
r = redis.Redis()
ps = r.pubsub()
ps.subscribe({"message"})
for msg in ps.listen():
    try:
        print(msg["data"].decode())
    except:
        pass
```

```
r = redis.Redis()
ps = r.pubsub()
ps.subscribe({"message"})
def main_loop():
    while True:
        msg = ps.get_message(ignore_subscribe_messages=True)
        if msg:
            print(msg["data"].decode())
        time.sleep(0.001)
threading.Thread(None,main_loop).start()
```

Redovi poruka

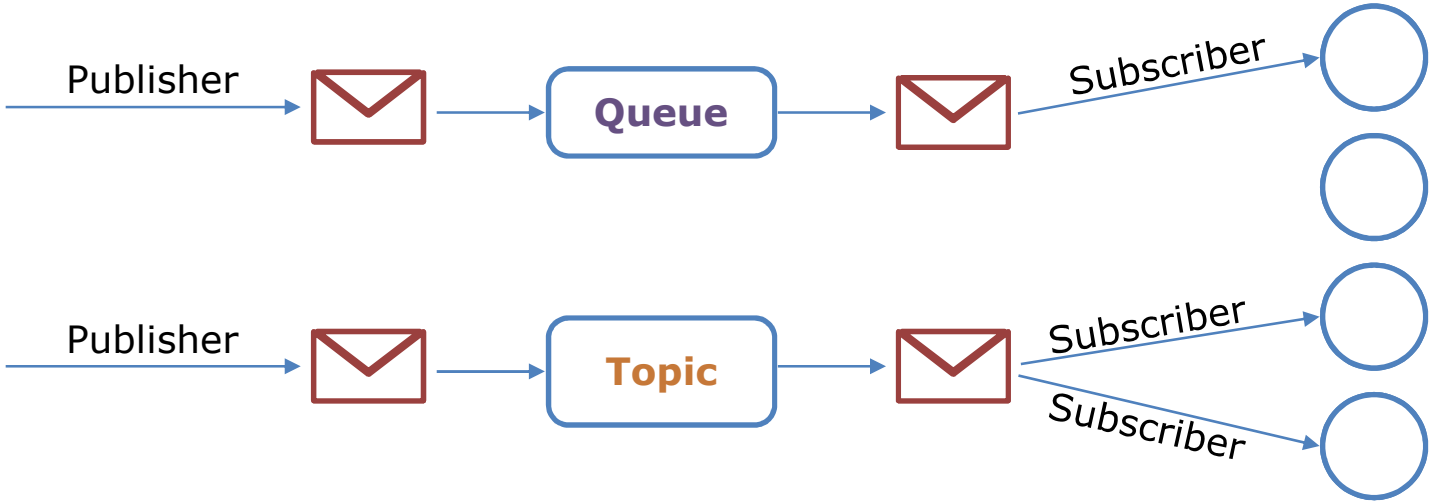
- Redovi poruka su aplikacije ili delovi aplikacija koji su u stanju da prihvataju, čuvaju i distribuiraju poruke

- Često se u svrhu redova poruka koriste namenski programi

ACTIVEMQ RabbitMQ ØMQ RocketMQ

Redovi i teme

- Dva osnovna načina distribucije poruka su red (**queue**) i tema (**topic**)
- Red uklanja dostavljenu poruku nakon prve uspešne isporuke
- Tema isporučuje poruku svim zainteresovanim učesnicima



Apache Active MQ

- ActiveMQ je popularan red poruka, otvorenog koda
- ActiveMQ redu poruka se može pristupiti mrežno nekim od protokola (REST, OpenWire, Stomp i drugi)
- ActiveMQ se može koristiti putem Python-a

pip install stomp.py

```
import stomp
```

Povezivanje i slanje poruka
(pdap-ex04 stomp)

- Povezivanje

```
conn = stomp.Connection()
conn.connect('admin', 'admin', wait=True)
```

- Slanje poruke

```
conn.send(body=str(time.time()), destination='/queue/results')
```

- Prekidanje konekcije

```
conn.disconnect()
```

Preuzimanje poruka
(pdap-ex04 stomp)

```
class MyListener(stomp.ConnectionListener):
    def on_error(self, headers, message):
        print('received an error "%s"' % message)
    def on_message(self, headers, message):
        print('received a message "%s"' % message)

conn = stomp.Connection()
conn.set_listener('', MyListener())
conn.connect(wait=True)

conn.subscribe(destination='/queue/results', id=1, ack='auto')
```

Relacione baze podataka

- Relacione baze podataka predstavljaju najpopularnije rešenje za opštu upotrebu i smeštanje podataka.
- U vreme intenzivnog razvoja i pomola sistema za smeštanje i upravljanje podacima, objektno orijentisano programiranje nije bilo previše zastupljeno. To je jedan od razloga što je razvoj sistema za upravljanje podacima iskoristio relacioni model
- Jezik kojim se podacima u ovakvim sistemima rukuje - SQL
- Relacione baze podataka su zrela tehnologija koja je dokazana u praksi
- Relacione baze podataka zasnivaju se na široko prihvaćenim standardima. Migracija između dva relaciona sistema za upravljanje bazama podataka ne predstavlja preveliku teškoću

MySQL

- Danas najkorišćeniji sistem za upravljanje relacionim bazama podataka je MySQL. MySQL je višenamenska relaciona baza, veoma popularna u raznim sferama programiranja. Brza je, ima veliki kapacitet i podržana je od strane svih važnijih operativnih sistema. Takođe, ova baza je besplatna.
- U nastavku ovog kursa biće korišćen isključivo MySQL kao sistem za upravljanje bazama podataka.

Tradicionalno korišćenje baze
(pdap-ex04 dbexample.py)

- Primer tradicionalnog korišćenja MySQL sistema za upravljanje bazama podataka iz Java programskog jezika biće demonstriran na primeru objekata klase Person.
- Za smeštanje objekata ovog tipa u bazu koristiće se baza podataka *test*, i u okviru nje tabela *person*. DDL te tabele izgleda ovako:

```
CREATE TABLE `test`.`person` (
  `person_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `age` INT NOT NULL,
  `pid` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`person_id`));
```

Upravljanje podacima

Konekcija

Unos

Ažuriranje

Čitanje

```
import mysql.connector as connector

db = connector.connect(host="localhost",database="test",passwd="",username="root")
cur = db.cursor()
cur.execute(
    "insert into person (name,age,pid) values (%s,%s,%s)",
    ('Peter',35,123)
)
db.commit()
insert_id = cur.lastrowid
print("Inserted user:",insert_id)
cur.execute(
    "update person set age=%s where person_id = %s",
    (36,insert_id)
)
db.commit()
print("Updated user:",insert_id)
cur.execute("select * from person")
for person in cur.fetchall():
    print(person)
db.close()
```

Credits

https://www.flaticon.com/authors/freepik

https://www.flaticon.com/authors/flat-icons

https://www.flaticon.com/authors/nikita-golubev

https://www.flaticon.com/authors/becris

https://www.flaticon.com/authors/catkuro